Measurements Should Generate Value, Rather than Data,

Niessink, F.; van Vliet, H.

published in
Proceedings of the Sixth International Software Metrics Symposium, Boca Raton, Florida, USA, November 4-6, 1999, IEEE, pp. 31-38.

1999

document version
Publisher's PDF, also known as Version of record

Measurements Should Generate Value, Rather Than Data *

Frank Niessink and Hans van Vliet

Division of Mathematics and Computer Science, Faculty of Sciences, Vrije Universiteit
De Boelelaan 1081, 1081 HV, Amsterdam, The Netherlands
Tel: +31 20 444 7781, Fax: +31 20 444 7653
E-mail: {F.Niessink, J.C.van.Vliet}@cs.vu.nl

Abstract

Success factors for measurement programs as identified in the literature typically focus on the 'internals' of the measurement program: incremental implementation, support from management, a well-planned metrics framework, and so on. However, for a measurement program to be successful within its larger organizational context, it has to generate value *for the organization. This implies that attention should also be given to the proper mapping of some identifiable organizational problem onto the measurement program, as well as the translation back of measurement results to organizational actions. In this paper, we present a generic process model for measurement-based improvement, which does cover the latter issues as well. We describe a number of common uses for measurement programs in software organizations, from which we derive additional 'external' success factors. In addition, we propose a number of activities that organizations can use to implement value-generating measurement programs.*

1. Introduction

Briand, Differding, and Rombach [2] state that 'Despite significant progress in the last 15 years, implementing a successful measurement program for software development is still a challenging undertaking.' We fully concur with this observation. Even if all of the commonly agreed upon success factors for measurement programs, as for instance identified in Hall and Fenton [7] are adhered to, the measurement program need not be a success. In our view, this is partly caused by the fact that these consensus success factors typically focus on the 'internals' of the measurement program. Success factors like incremental implementation, support from management, a well-planned metrics framework, and the like, are aimed at ensuring an ongoing flow of proper data. However, for a measurement program to survive in the long run, it should also be successful within its larger organizational context. It has to generate *value* for the organization. This requires that attention be given to the proper mapping of some identifiable organizational problem onto the measurement program, as well as a translation back of measurement results to organizational actions.

In this paper, we present a generic process model for measurement-based improvement which does cover the latter issues as well. This generic model is described in section 2. In section 3 we show that the consensus success factors for measurement programs as identified by Hall and Fenton only cover part of our measurement-based improvement model. We then describe a number of common uses of measurement programs in section 4. From those uses we derive a number of success factors that are external to measurement programs. Next, in section 5, we propose a number of steps organizations could take to fulfill the success factors identified in section 4. Finally, section 6 presents our conclusions.

2. Modeling measurement-based improvement

Figure 1 displays a generic process model for measurement-based improvement. It more or less resembles a 'pretzel', a loaf of bread in the form of a loose knot[1]. The pretzel consists of two parts—the two halves, three concepts—the black dots, and four steps—the four arrows.

The cycle starts with an organizational problem or goal (left black dot). We do not assume anything about the 'size' of the problem or goal. A problem could only affect one developer or the whole organization, in both cases the same steps have to be passed through. The organization analyses the problem (upper left arrow), and arrives at one or

*Proceedings of the Sixth International Software Metrics Symposium, November 4-6, 1999, Boca Raton, Florida, USA, pp. 31-38.

[1] Of course, from a mathematical point of view, the figure looks like a lemniscate of Bernoulli.

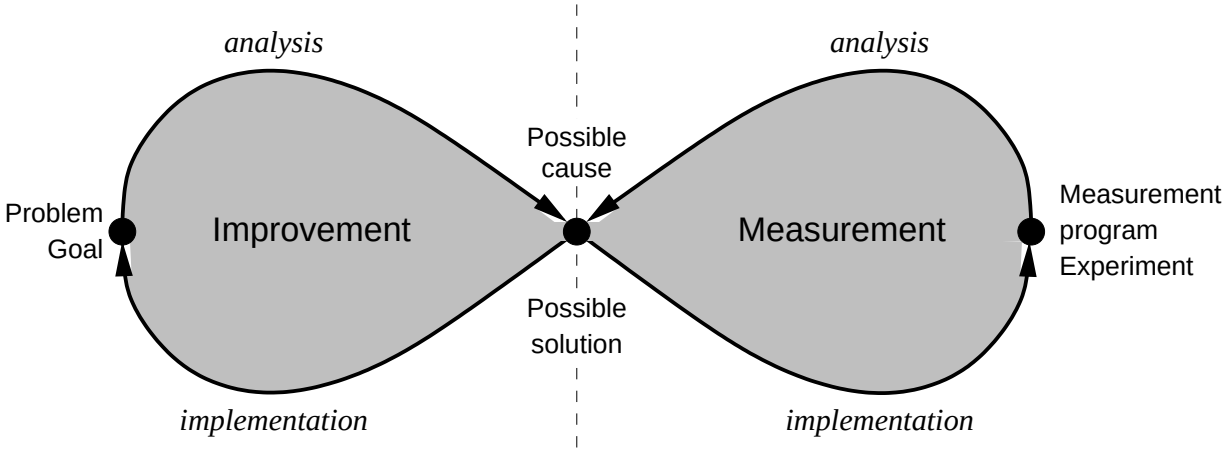

Figure 1. A generic process model for measurement-based improvement

more possible causes of the problem and/or possible solutions (middle dot). The analysis will generally be based on a combination of knowledge about the own organization, knowledge from literature ('theory'), and common sense. Next, the organization has to decide whether it has sufficient knowledge to establish the cause of the problem and correct it, or to reach the stated goal. If this is the case, the organization need not traverse the right cycle. In most cases, however, the organization needs to find out which of the possible causes is the real cause of the problem, or which of the possible solutions is the best solution. In addition, it may need extra information to implement the solution. To gather this information, the organization can design an experiment or set up a measurement program (lower right arrow). Executing the measurement program or experiment (right dot) results in the gathering of data, which is analyzed and related to the problem or solution at hand (upper right arrow). Finally, the organization solves the problem or reaches the goal by implementing the solutions found (lower left arrow).

Although both the preceding description and the arrows in figure 1 suggest a chronological sequence of steps, this is not necessarily the case. The arrows merely indicate causal relations. Hence, the model does not prescribe a single loop through the lemniscate. It is very well possible for an organization to iterate the right loop a number of times before implementing a solution. For example, it may be necessary to first implement an experiment to find the cause of a problem, and then implement another experiment to find a suitable solution. Moreover, organizations might also want to implement a solution and a measurement program in parallel, to monitor the implementation of the solution.

Let us illustrate the model by means of an example, see

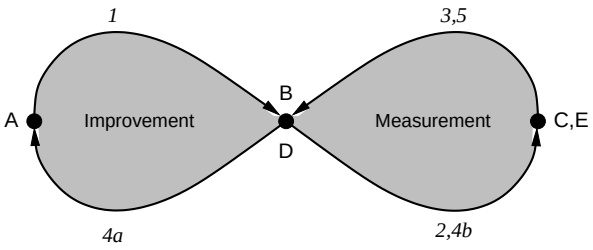

Figure 2. An example of measurement-based improvement

figure 2. Suppose a software maintenance organization has problems planning the implementation of change requests. Often, the implementation of specific change requests takes much more time than planned, and the organization fails to deliver the changed software in time. So, the problem this organization faces is the inaccurate planning of change requests (A). After analyzing the problem (*1*), the organization discovers that it does not know which factors influence the time needed to implement change requests (B). The organization decides to investigate this, and designs (*2*) a short-running measurement program (C) to investigate possible factors. After running this measurement program for a limited period of time, the gathered data are analyzed (*3*). We assume that a number of factors are found that influence the effort needed to implement change requests (D). Next, a planning procedure is developed and implemented (*4a*) in which the factors found are used to estimate the effort

needed to implement the change requests. An accompanying measurement program (E) is designed (*4b*) to gather the data needed for the new planning procedure and to monitor the accuracy of the planning (*5*).

We conclude this section with a few remarks on the nature of the presented generic process model of measurement-based improvement.

First, one could wonder whether this model is prescriptive or descriptive. We assume that if software organizations want to improve their processes or products, and use measurement to support those improvements, they will perform the activities as we have described above. That means we use the model as a representation – though very abstract – of what goes on in reality; i.e. it is a descriptive model. One could argue that the model is also a prescriptive model; it tells us which activities to perform when conducting measurement-based improvement. However, because of the high level of abstraction, the model is unsuitable to directly support organizations in their measurement-based improvement efforts.

Second, the model resembles the Goal-Question-Metric paradigm [1]. One could be tempted to map the GQM goal on the left black dot, GQM questions on the middle dot, and the GQM metrics on the right dot. However, the goal of the GQM-paradigm and the goal of the process model are not the same: the goal in the pretzel is an organizational goal, whereas the goal in the GQM-paradigm is a measurement goal. Still, GQM can very well be used to support the design of the measurement program (lower right arrow). Adaptations of GQM, such as described in [10, 16], focus on the right side of the pretzel as well.

Third, the distinction made in the model between improvement on the one hand, and measurement on the other hand, corresponds with the distinction made by Kitchenham, Pfleeger, and Fenton [9] between the empirical, real world and the formal, mathematical world. Their structural model of software measurement consists of two parts: an empirical world and a formal world. The empirical world contains entities that can have certain properties, called attributes. The formal world consists of values that measure the attributes of entities, expressed in certain units. Measurement now, is the mapping of a particular entity and attribute from the real world to a value in the formal world. The generic process model reflects the differences between these two worlds: measurement activities (the right half) are concerned with constructing a formal world based on the real world, whereas improvement activities (the left half) are concerned with changing the real world based on the formal world created by the measurement activities.

3. Success factors for measurement programs

In this section we use the generic process model described in section 2 to illustrate the scope of the set of success factors identified by Hall and Fenton [7]. In [15] we have also mapped a number of other frameworks for measurement programs [5, 4, 2, 13] onto our model. The results thereof are very similar.

Hall and Fenton [7] identify a number of consensus success factors for the implementation of measurement programs. Table 1 shows these factors, that were identified after studying other literature, such as [6, 17]. A closer look at the success factors shows that they are mainly targeted at reducing the risk of failure. For example, the motivation given by Hall and Fenton for factor six – usefulness of metrics data – is not that the measurement program should have added value for the organization, but rather that the usefulness should be obvious to the practitioners. From the 15 success factors, 10 are targeted at gaining the acceptance of the practitioners involved (4-9, 11, 13-15). The other five factors are concerned with reducing the risk of failure by advocating a gradual introduction and improvement of the program. The measurement program should be incrementally implemented, constantly improved, use existing materials, be supported by management, and a well-planned metrics framework should be used (1-3, 10, 12).

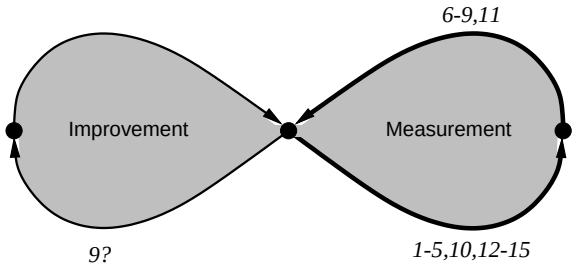

Figure 3. The success factors mapped to the generic process model

Figure 3 shows how the success factors can be mapped onto the generic process model. The majority of the success factors mentioned by Hall and Fenton refer to the implementation of measurement programs. Some are concerned with the collection and analysis part, and only one success factor is concerned with the usage of the measurement data (factor nine). That factor is marked with a question mark, because Hall and Fenton motivate it in terms of acceptance of the measurement program by the practitioners, rather than in terms of added value of the program to the company.

1	Incremental implementation
2	Well-planned metrics framework
3	Use of existing metrics materials
4	Involvement of developers during implementation
5	Measurement process transparent to developers
6	Usefulness of metrics data
7	Feedback to developers
8	Ensure that data is seen to have integrity
9	Measurement data is used and seen to be used
10	Commitment from project managers secured
11	Use automated data collection tools
12	Constantly improving the measurement program
13	Internal metrics champions used to manage the program
14	Use of external metrics gurus
15	Provision of training for practitioners

Table 1. Consensus success factors

4. Possible uses for measurement programs

Measurements should generate value to the organization. This value is determined outside the measurement program proper. A major factor determining the success of a measurement program is whether or not it actually does create that value. In the previous section we have shown that the measurement program success factors listed by Hall and Fenton [7] focus on the measurement program internals. In this section, we investigate different situations in which a measurement program may be used to gather data needed to solve organizational problems or help reach organizational goals. The purpose is to derive additional success factors, external to the measurement program, but nevertheless essential for the success of the measurement program.

Measurement programs can serve many purposes, and hence create value to a software organization in different ways. In our experience, the main kinds of purposes for which measurement programs are used are:

- **Reporting** A situation where there is a contractual obligation to reach certain targets. For example, a software maintenance organization may guarantee in its service level agreements some level of availability of a system, or some maximum down-time. The actual performance of the organization is then monitored, and results are reported to the customer. Often, the agreement explicitly states penalties incurred in case of non-fulfillment of the agreement.

  What measurements need to be taken for reporting purposes can fairly easily be derived from the service level agreement at hand. However, the measurement program's value can be improved by not only measuring the service levels covered by the service level agreement, but also factors that enable the organization to predict situations that might cause the service levels to be violated. For example, if a service level agreement includes a threshold on the maximum response time of certain information systems, the service provider might want to measure the load of the server that runs the software. That way the service provider can prevent high response times by keeping the server load low enough.

- **Monitoring performance** In this situation, someone (usually management) sets the standards, usually in terms of a set of performance indicators, and measurements serve to see whether these levels of performance are met. The main difference with the reporting case is that the 'customer' of the data is external in the reporting case, while it is most often internal in this case.

  It is vital for the success of this kind of measurement program that the organization has a clear plan on how to act if the desired performance is not being achieved. For example, if the organization wants to measure schedule slippage, it also needs to be prepared to take measures to improve schedule and planning accuracy. The latter type of measure is often not dealt with explicitly. As a result, the organization is likely to play the ostrich in case expectations are not met.

- **Learning** The organization has a problem but does not immediately see a solution to it. First, it needs to investigate the problem more thoroughly, and find the root causes, or main underlying factors, that cause the problem.

  For example, a software maintenance organization performs corrective maintenance for a large number of

customers for a fixed price per period. It needs to be able to estimate the corrective maintenance workload (i.e. the expected number of bugs) to be able to set a reasonable price. The software maintenance organization starts a measurement program to identify the main factors that determine the corrective maintenance workload. If those factors are found, the organization could use this information in the form of an estimation procedure to support the bidding process for new contracts. Probably, the organization will also want to keep monitoring both the factors and the actual corrective maintenance workload for the different contracts in order to calibrate the estimation procedure.

- **Performance improvement** In this case, certain relations between (product and/or process) variables are assumed or formulated. For example, a software development organization assumes that the later bugs are fixed during the development process, the more expensive the fix is. The organization decides to strive for phase containment of faults [8]. A measurement program is then started to gather the necessary data. Next, the data are analyzed, and actions are taken based on the outcome of the analysis. For example, measures can be taken to improve the in-phase detection of faults. This process usually is a cyclic one, whereby hypotheses get formulated, refined or rejected, and new hypotheses guide the next cycle.

- **Organizational health** This is kind of a check-up. The organization is compared against a set of norms (usually created externally). In this case, it is most interesting to consider the case where the norms are *not* met. What kind of action, if any, will be taken in that case? And how does the check-up help in deciding what the best actions would be? In the case of an assessment of the software process against a set of norms like put down by the Software CMM [3, 11], the assessment results in a list of recommendations for improvements. In the case of a benchmark against industry averages, the actions that should be taken as a result of the comparison are less clear.

- **Navigation** In this situation, management determines a destination, or at least a direction for travel. Next, a plan is made how to get there. During the subsequent journey, measurements are used to answer questions like ”How well am I following the plan?”, ”Have I reached my destination yet?”, or ”Was the journey worth it?”. Again, it is generally worthwhile to pay special attention to cases where the answer to these questions is negative.

From this list of typical applications of measurement programs, four success factors – external to the measurement program – emerge:

1. Various assumptions underlie the measurement program. These assumptions should be made explicit and it should be decided if and when these assumptions are tested. These assumptions often take the form of a cause-effect relation between anticipated changes and a desired result.

2. Different outcomes can result from a measurement program. An organization should consider all possible – negative and positive – outcomes and decide how to act on them. Often, only one of these possible outcomes is satisfactory: performance is ok, targets are met, etc. It is the other possible outcomes that are most interesting from our point of view: what happens if the performance is not ok, targets are not met, etc. If it is not specified what to do in those cases, there is quite a chance that *nothing* will be done.

3. The organization should act according to the outcomes of the measurement program, in order to reach the goals set or solve the problems identified. This applies to both negative and positive outcomes. If the organization does not act, the value of the measurement program degrades, and it will sooner or later, but usually sooner, come to an end.

4. The organization should monitor the changes implemented, in order to verify that these changes indeed constitute an improvement for the organization. Measurement involves modeling, and thus abstracting away from many aspects. We should verify that our model captures reality sufficiently well, and keeps doing so if reality changes over time. Also, it should be verified whether the desired outcome is brought about (by the changes implemented or for any other reason).

In the next section, we propose a number of activities that organizations can follow to fulfill the success factors described above.

5. Steps for measurement-based improvement

In this section we describe steps an organization could take to fulfill the success factors identified in the previous section. These steps are illustrated using an example, based on a measurement program described elsewhere [12].

1. Determine the valuable outcome of the measurement and/or improvement program. The organization in question explicitly determines what results it expects from the measurement and/or improvement program and how these results are to be measured.

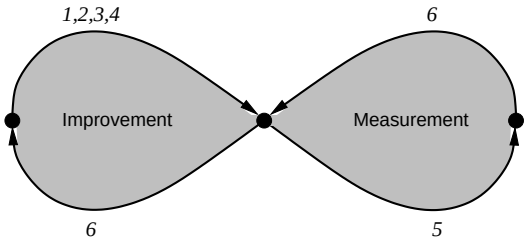

Figure 4. Steps for measurement-based improvement

For example, a software maintenance organization and its (only) customer have difficulties determining a fair price for change requests. Together, the software maintenance organization and the customer decide to implement function points – more specifically maintenance function points – as a means to determine the price of change requests. Hence, the valuable outcome of this improvement initiative is to have an objective mechanism to determine a fair price for change requests.

2. Assumptions about relationships between changes to be made and results to be obtained are made explicit.

   Note how the organization assumes that: (1) the maintenance function points will indeed be an objective measure of the volume or size of a change request, and (2) the number of maintenance function points of change requests are correlated with the effort needed to implement those changes, which is needed for a reasonable fair price.

3. Develop a plan to obtain this outcome. Improvement can only be achieved by changing the organization in one or more respects. The plan determines how and what is going to be changed. It is decided which assumptions need to be tested before the changes are implemented, and which are checked during or after implementation of the changes. Hence, this step also results in the measurement goals to be fulfilled by the measurement program.

   In our example, the software maintenance organization and the customer design a new change request planning procedure which includes the counting of maintenance function points of each change request to determine its price. Because this is the first time this particular function point model is being used, it is decided to use the model for a while and then analyze its behavior. Specifically, correlation with the effort needed to implement the changes will be investigated.

   So, the measurement program to be implemented needs to fulfill two requirements: (1) provide the necessary information to apply the maintenance function points, and (2) provide the information needed to evaluate the maintenance function point model.

4. Follow-up scenarios are developed. For each possible outcome of the measurement program, scenarios are developed that describe how to act on that particular outcome.

   If the correlation between maintenance function points and effort is lower than a certain value, the model structure will be adjusted. For example, the model makes certain assumptions about the cost of deleting functions: it states that deletion of a function costs 20% of the effort needed for building the function. If needed, that factor of 0.2 can easily be adjusted using the effort data.

This completes the first phase (the upper-left arrow) of the measurement-based improvement process model. One way to continue from here is to set up a measurement program, analyze its results, and only then implement some changes. In that case, we apparently are not quite sure yet whether the assumptions made in the previous steps really hold. In case we are very confident about these assumptions, we may decide to just implement the changes, and not bother about measurements at all. An intermediate form is to do both at the same time: some changes are implemented, and at the same time a measurement program is started to be able to do an a posteriori check on the viability of those changes. In general, it depends on the situation at hand which of these continuations is to be chosen. In the example we are considering here, it is reasonable to follow the last one identified. So we will start to use function points as an objective effort measure, and at the same time we start a measurement program in order to be able to test our function point model.

5. Design and implement the improvements and the measurement program.

   The new planning procedure is implemented. The measurement program to gather the function points and the effort data is implemented. The organization develops a detailed measurement protocol and counting guidelines. The measures to be taken are the input data for the function points, i.e. data element types, record element types, etc., and the effort data per function changed, needed for the evaluation.

6. Act upon the measurement program (step 5) according to the scenarios developed in step 4.

   After a while, the measurement data are analyzed and, depending on the outcome, one of the scenarios de-

veloped in step 4 is executed. In this case, the function point model assumes that the size of a function changed and the size of the change itself contribute equally to the effort needed for the change (i.e. changing a function of size $2x$ takes twice as much effort as changing a function of size x and changing 60% of a function takes twice as much effort as changing 30% of a function). However, the analysis shows that the size of the function is much more important for determining the effort than the relative portion of the function that is changed. Hence, the function point model is changed to reflect these findings, and the procedures are adapted to the new version of the model.

The steps listed are an example of how an organization could implement measurement-based improvement, making sure that the success factors described in the previous section are taken into account.

Note that though the example as described above is fictional, it is based on a real measurement program [12]. In reality, the organization did not make the assumptions listed in step 2 explicit. We were asked to analyze the function point model. The fact that we were indeed able to analyze it was a mere coincidence: the effort data was for a large part recorded on the level of changes to individual functions, where it could have been recorded at a more coarse level of granularity just as well.

When we discovered that the model needed structural changes to improve its correlation with effort, the organization was not prepared to make those changes. One of the reasons was the fact that the organization was rather happy with the model, despite the low correlation, because it solved part of the problem, i.e. it provided an objective pricing mechanism for change requests. The fact that the function model could need calibration was not explicitly recognized up front. A commitment to act upon the outcome of the measurement program was not made. Not surprisingly, our findings were not implemented, and things stayed as they were before our analysis of the function point model.

Two other measurement programs that we carried out obeyed the same (large) subset of the Hall and Fenton's success factors [14]. Nevertheless, the usefulness of their results differed widely. In both programs, we had not explicitly dealt with the steps as identified above. On hindsight, we found that the assumptions underlying much of our measurements were (accidently) valid in one case, and not valid in the other.

6. Conclusions

In this paper we have introduced a generic process model for measurement-based improvement. We have used this model to show that the consensus success factors for measurement programs, such as the ones presented by Hall and Fenton, focus on the internals of measurement programs. We have argued that to guarantee the success of measurement programs, one should also take external factors into account. These external factors are aimed at making sure that the measurement program generates value, and not just data. By discussing different uses of measurement programs we have identified four external success factors of measurement programs:

1. The various assumptions underlying the measurement program should be made explicit. It should be decided if and when these assumptions are tested.

2. Different outcomes can result from a measurement program. An organization should consider all possible – negative and positive – outcomes and decide how to act on them.

3. The organization should act according to the outcomes of the measurement program, in order to reach the goals set or solve the problems identified.

4. The organization should monitor the changes implemented, in order to verify that these changes indeed constitute an improvement for the organization.

Our external success factors complement the success factors as presented by Hall and Fenton. Together, these success factors cover all four phases of the measurement-based improvement process model as presented in section 2.

We have shown how an organization could adhere to the external success factors by explicitly addressing these issues before designing and implementing a measurement program. A preliminary assessment of various measurement programs conducted by us shows that this model provides a good vehicle to explain important differences in the success of these programs. We are currently in the process of further assessing this model in industrial settings.

Acknowledgments

This research was partly supported by the Dutch Ministry of Economic Affairs, project 'KWINTES', nr. ITU-96024. Partners in this project are Cap Gemini, Twijnstra Gudde, the Tax and Customs Computer and Software Centre of the Dutch Tax and Customs Administration, and the Technical Universities of Delft and Eindhoven.

We especially thank Rob Donnellan of Cap Gemini, USA, for the many fruitful discussions we had on the topic of this paper, and for sharing with us his experiences in setting up value-generating measurement programs in actual practice.

References

[1] V. R. Basili and H. D. Rombach. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, June 1988.

[2] L. C. Briand, C. M. Differding, and H. D. Rombach. Practical Guidelines for Measurement-Based Process Improvement. *Software Process – Improvement and Practice*, 2(4):253–280, Dec. 1996.

[3] Carnegie Mellon University/Software Engineering Institute. *The Capability Maturity Model: Guidelines for Improving the Software Process.* SEI Series in Software Engineering. Addison-Wesley Publishing Company, 1995.

[4] P. Comer and J. Chard. A measurement maturity model. *Software Quality Journal*, 2(4):277–289, Dec. 1993.

[5] M. K. Daskalantonakis, R. H. Yacobellis, and V. R. Basili. A Method for Assessing Software Measurement Technology. *Quality Engineering*, 3:27–40, 1990-1991.

[6] R. B. Grady. *Practical software metrics for project management and process improvement.* Hewlett-Packard Professional Books. Prentice-Hall, Inc., 1992.

[7] T. Hall and N. Fenton. Implementing Effective Software Metrics Programs. *IEEE Software*, 14(2):55–65, March/April 1997.

[8] A. Hevner. Phase containment metrics for software quality improvement. *Information and Software Technology*, 39(13):867–877, Dec. 1997.

[9] B. Kitchenham, S. L. Pfleeger, and N. Fenton. Towards a Framework for Software Measurement Validation. *IEEE Transactions on Software Engineering*, 21(12):929–944, Dec. 1995.

[10] F. v. Latum, R. van Solingen, M. Oivo, B. Hoisl, D. Rombach, and G. Ruhe. Adopting GQM-Based Measurement in an Industrial Environment. *IEEE Software*, 15(1):78–86, January/February 1998.

[11] B. McFeeley. IDEAL^{SM}: A User's Guide for Software Process Improvement. Handbook CMU/SEI-96-HB-001, Software Engineering Institute/Carnegie Mellon University, Feb. 1996.

[12] F. Niessink and H. van Vliet. Predicting Maintenance Effort with Function Points. In M. J. Harrold and G. Visaggio, editors, *International Conference on Software Maintenance*, pages 32–39, Bari, Italy, October 1-3, 1997. IEEE Computer Society.

[13] F. Niessink and H. van Vliet. Towards Mature Measurement Programs. In P. Nesi and F. Lehner, editors, *Proceedings of the Second Euromicro Conference on Software Maintenance and Reengineering*, pages 82–88, Florence, Italy, March 8-11, 1998. IEEE Computer Society.

[14] F. Niessink and H. van Vliet. Two Case Studies in Measuring Software Maintenance Effort. In T. M. Khoshgoftaar and K. Bennett, editors, *International Conference on Software Maintenance*, pages 76–85, Bethesda, Maryland, USA, November 16-20, 1998. IEEE Computer Society.

[15] F. Niessink and H. van Vliet. A Pastry Cook's View on Software Measurement. In R. Dumke and A. Abran, editors, *Software Measurement - Current Trends in Research and Practice*, pages 109–125, Wiesbaden, Germany, 1999. Deutscher Universitäts Verlag.

[16] R. Park, W. Goethert, and W. Florac. Goal-Driven Software Measurement – A Guidebook. Technical Report CMU/SEI-96-HB-002, Software Engineering Institute/Carnegie Mellon University, 1996.

[17] S. L. Pfleeger. Lessons Learned in Building a Corporate Metrics Program. *IEEE Software*, 10(3):67–74, May 1993.